Defense Advanced Research Programs Agency: Study

System Utility Function for Adaptive/Reconfigurable MANETs[*]

Gustavo de Veciana
Email: gustavo@ece.utexas.edu

1 Goals and Discussion

In this project a system *utility function* for a MANET will play a dual role. Its first role is as a means to specify *high-level objectives.* By associating values to various possible system states the system utility function serves to drive adaptation/reconfiguration mechanisms towards better meeting the demands of a specific mission or dynamic set of applications. In other words it implicitly captures the intent/tradeoffs the network designer would make rather than specifying particular configurations for each possible operational regime. Capturing such objectives via a single overall utility function is itself a nontrivial goal, particularly as there is a need to do so over a wide set of possible scenarios. Furthermore for mission-oriented applications the *tactical context* will play a critical role. Similarly in commercial networks one would expect business, security and other requirements to have a critical impact on the overall system utility. Thus designing a utility function capturing mission goals will involve rules mapping specific contexts, e.g., an emergency or high risk period, to appropriate utilities/sensitivities among ongoing tasks. In addition to specifying an overall utility it would be useful to independently supplement it by introducing further additional goals, or *operational constraints*, e.g., connectivity requirements, network/node lifetime requirements, additional fairness or strict prioritization among application types. Operational constraints should be viewed as explicit actions and/or goals that take precedence over optimization of the system utility function. The second role of the system utility function is as a *test and evaluation* metric to assess the effectiveness of reconfiguration/adaptation mechanisms. In this study we will make an initial recommendation for the basic building blocks of a system utility function for MANETs, discuss some possible complementary operational constraints, and briefly consider interface requirements. We conclude with some challenges faced in defining an overall utility function for MANETs.

2 Recommendation

2.1 Tactical MANET: Application types and task workloads.

We start by briefly discussing a basic set of application types that would be used in a tactical mission which leverages a MANET. The descriptions are perforce high level recognizing that new applications are

[*]Approved for Public Release. Distribution Unlimited.

emerging. The goal is to give a fairly clear picture of the character of the workload that will be supported by tactical MANETs.

File transfers, e.g., files with maps or slides, to a set of nodes/users are likely to be mediated via a reliable multicast service. End nodes are likely to be sensitive to file transfer delays, i.e., not the delay of individual packets but the time to reliably receive the full document. Alternatively such tasks are sensitive to the bandwidth they are allocated throughout their lifetime in the system. If multiple receivers are involved, the task may be sensitive to the delay until all have the file. For a typical reliable multicast transport mechanism the delays to complete the transfer to various receivers would be roughly the same, i.e., are bottlenecked by the slowest receiver.

Situational awareness (SA) applications keep each node/user associated with a multicast session appraised of the current location and state of other nodes/users. Thus each node/user transmits update packets at regular intervals, e.g., 1 sec intervals, a fraction of which should arrive within some delay, e.g., 5 sec, so that each node can maintain a consistent picture of the overall state. Update packets that are excessively delayed, e.g., 5 sec, or packets that arrive out of order are deemed worthless. SA applications would likely be built on an unreliable multicast transport mechanism. SA applications are typically ongoing throughout a mission, though their 'importance' may vary over different phases of the mission.

Packetized voice sessions, e.g., based on VoIP technology, will be one of the important basic application types. For tactical communications these are likely to be of the *push to talk* type, i.e., users join a multicast session and talk to the set via a push to talk protocol. The traffic volume associated with voice would be fairly high, when users are active, and fairly bursty, i.e., a user will go from the active to listen/standby states and the active user in a group may change around in a correlated fashion – i.e., only one user is active at a time. Such applications would use unreliable multicast transport since they are sensitive to packet delays and retransmitting lost packets is not likely to be useful. Note reliable multicast service might also be used if longer delays and high jitter can be smoothed out by using playback buffers and/or the session is not interactive. Multiple, possibly overlapping, voice sessions may be ongoing over different sets of users, for fairly long periods of time.

Real-time packet video involves a flow of packetized image and audio signals carried across the network to one or more end points. As with voice, packets should preferably not be dropped and are not useful if they are excessively delayed. The quality of service seen by end users for video will depend on a variety of features including how traffic is coded and packetized, e.g., layered encoding, priority sensitive packetization, forward error correction etc. For simplicity, we will assume that quality improves with the fraction of packets that arrive without being excessively delayed to the set of receivers. Real-time packet video would typically be built upon an unreliable multicast or RTP/UDP transport service permitting flexible one-to-one or one-to-many video transmissions.

Collaborative/whiteboard applications involve a group of end nodes maintaining a common state, i.e., the state of the shared whiteboard. Each node/user may generate commands that must be shared on a timely basis with all nodes/users involved in the session so that they have a common view of the whiteboard. This is similar to situational awareness applications, except that the traffic generated by nodes/users might be more bursty. Such applications might further involve file transfers, voice, video, etc., which we consider as separate tasks.

General observations on tactical mission application types and ad hoc networking. Our first observation is that although tactical missions may involve point-to-point sessions, the common case is one involving *sets* of users exchanging information via reliable/unreliable multicasting. Thus the overall utility

character	*application type*	*transport mechanism*	*QoS sensitivities*
elastic	file transfer	reliable multicast/TCP	file transfer delay
inelastic	situational awareness	unreliable multicast	lost
	voice	unreliable multicast	and/or
	video	unreliable multicast/RTP-UDP	excessively
	collaborative	reliable multicast	delayed packets

Table 1: Table exhibits the basic application types, associated transport mechanism and main quality of service attributes of interest.

of the network in supporting a given task will need to integrate the perspective of a *set* of receivers. The second observation is that the resources involved in supporting communications over an ad hoc network may vary dramatically based on the network topology and location(s) of end node/user(s). Indeed, connectivity is maintained via hop-by-hop relaying, thus nodes that are far from each other will require more resources to communicate than those that are close by. When bandwidth and/or energy are limited the overall utility function will need to reflect reasonable tradeoffs among tasks that are of similar importance but have dramatically different resource requirements. For example, in some cases one may wish to be fair among such tasks, whereas in others one may wish to discriminate among tasks due on the scarce resources required to support them. Finally we note that the above application types exhibit different sensitivities to the end to end performance they see, which in turn also depend on the transport mechanism upon which the application is built. Table 1 summarizes the application types and quality of service aspects for the key applications we have introduced. As shown in the table we differentiate between *elastic* applications which are sensitive to the perceived transfer delay, or average bandwidth, and *inelastic* applications that are sensitive to lost and/or excessively delayed packets over time.This will be discussed in more detail in the sequel.

Independent task workloads. We shall define an *independent* task workload for a MANET, as a sequence of tasks that are offered to the network during a given period – this is defined in more detail below. Independence refers to the that fact that the *same* workload is offered to the network irrespective of whether one or more of the tasks in the sequence fails to complete adequately. If tasks do exhibit such dependencies, they could in principle be grouped into *task threads*, i.e., dependent sequences of tasks, and workloads consisting of independent task threads might be considered. For now we focus on independent task workloads and leave the problem of considering task threads for future work – see Section 5.

2.2 Current and overall system utility functions

We model the overall evolution of the system using discrete time slots indexed $t = 1, 2, \ldots$ where each slot has a duration Δ secs. We model an *independent* task workload for a MANET as a sequence of tuples

$$W = \{(a_i, R_i, s_i, u_i, w_i) | \ \ i = 1, 2, \ldots, m\}.$$

Here $a_i \in T$ denotes the application type for task i, and T is a set of possible application types, e.g., voice, file transfer, etc. The set R_i corresponds to a set of receiver nodes/users associated with task i, e.g., the members of a multicast group. Thus if the task is a point-to-point unidirectional session R_i might contain a single receiver node. If task i corresponds to a sender transmitting a file to a set of users, then R_i would contain only the set of receivers for that file. If the task involves a set of nodes, which are both receivers and senders, e.g., push to talk voice, situational awareness, then R_i would contain all nodes participating as both receivers and senders. Finally $s_i \in \mathbb{Z}^+$ the start time for task i. The last two elements of the tuple u_i and w_i correspond to a utility function and weight associated with task i and will be explained in more detail below.

notation	*meaning*
W	task workload
i	index for tasks $i = 1, \ldots m$
a_i	type of task i where $a_i \in T$ is one of a set T of possible types
s_i	start time for task i where $s_i \in \mathbb{Z}^+$
R_i	set of receivers associated with task i
f_i	finish time time for task i where $f_i \in \mathbb{Z}^+$
A_t	set of tasks that are active at time t
$u_i()$	utility function $u_i : \mathbb{R}^{\lvert R_i \rvert} \rightarrow \mathbb{R}$ associated with task i
w_i	weight associated with task i where $w_i \in \mathbb{R}^+$
$\vec{q}_i(t)$	QoS vector $\vec{q}_i(t) = (q_i(t,r) \lvert r \in R_i)$ for task i's receivers at time t

Table 2: Table summarizes notation associated with describing task workloads

An adaptation/reconfiguration policy p may affect both the time at which a task i finishes, denoted f_i, and the quality of service task i sees on each time slot during its sojourn in the system. At any time t we let A_t denote the set of tasks that are active in time slot t, i.e.,

$$A_t = \{i | s_i \leq t \leq f_i, i = 1, \ldots, m\}.$$

For each task which is active at time t, i.e., $i \in A_t$, we let $\vec{q}_i(t) = (q_i(t,r) | r \in R_i)$ denote a vector capturing the quality of service the network delivers to task i's receivers during slot t. Note that $\vec{q}_i(t)$ may correspond to different quality of service metrics depending on the task type. Note that although we have not denoted this explicitly a reconfiguration/adaptation policy p may affect the the finishing time of tasks, and thus the active set at a particular time, as well as each task's quality of service vector.

We shall define an additive utility function for the MANET based on two quantities associated with each task. We let $u_i(\vec{q}_i(t))$ be a function that captures the 'utility' that task i derives from the QoS vector during time slot t. We shall assume that $u_i(\cdot)$ is normalized so that it is at most 1 corresponding to the best possible QoS. If the utility is 0, this corresponds to a task which is given insufficient resources, i.e., is essentially blocked. In the next section we will associate different types of utility functions to tasks depending on the application types. We further associate a weight w_i with each task i intended to capture its relative importance within the mission at hand. Note that in this framework the weight associated with a task is static and preassigned by the mission designer or assigned on the fly by a 'mission understanding' subsystem. It may make sense, in some cases, to have a 'task' that is ongoing over different phases of a mission have different weights during those phases. To capture this, we would include in our workload a sequence of identical tasks which abut on each other but have different weights.

Additive current weighted utility function. We define the overall utility $U_t(W, p)$ for a MANET under workload W and policy p at time t by

$$U_t(W, p) = \sum_{i \in A_t} w_i \, u_i(\vec{q}_i(t)) \qquad \text{(additive overall utility function at time } t\text{)} \tag{1}$$

and the normalized overall utility at time t by

$$U_t^n(W, p) = \frac{\sum_{i \in A_t} w_i \, u_i(\vec{q}_i(t))}{\sum_{i \in A_t} w_i} \qquad \text{(normalized additive utility function at time } t\text{)}. \tag{2}$$

Thus the normalized additive utility at time t can be at most 1. Note if A_t is empty we shall follow the convention that $U_t^n(W, p) = 1$.

Overall normalized system utility function. Suppose the mission lasts t^* time slots. We define the overall system utility $U^n(W,p)$ for a MANET subject to workload W and adaptation/reconfiguration policy p as the *time-average* over the period of operation, i.e.,

$$U^n(W,p) = \frac{1}{t^*}\sum_{t=1}^{t^*} U_t^n(W,p) \qquad \text{(overall normalized system utility).} \tag{3}$$

With the above conventions if a system achieves an overall normalized system utility close to 1 then the adaptation/reconfiguration policy has realized optimal tradeoffs with respect to the weights and task utility functions the system designer provided.

2.3 Utility functions for different application types

We shall begin by considering basic elastic and inelastic point-to-point tasks, i.e., tasks where the set of receivers R_i includes only one node which we denote by r, i.e., $R_i = \{r\}$, and so the QoS vector is in fact a scalar, i.e., $\vec{q}_i(t) = (q_i(t,r))$. Subsequently in Section 2.3.2 we consider ways of composing these to handle tasks that include multiple receivers.

2.3.1 Application-level utility functions for point-to-point sessions

Application-level utility functions–elastic case. We say that a task i is *elastic* if its utility on a given time slot depends on the *bandwidth* seen at the receiver r. Specifically we let $q_i(t,r)$ be

$$q_i(t,r) = \frac{\text{bits successfully received by } r \text{ on time slot } t}{\Delta} \tag{4}$$

i.e., the receive bandwidth for time slot t. We define the utility of an elastic task as an *increasing concave* function of $q_i(t,r)$. Two possible options are shown in Figure 1. The simplest alternative is to use a linear function

$$u_i(x) = \frac{x}{p_i},$$

where p_i denotes the peak receive rate for r or an upper bound for it. In this case the utility of an elastic task on a given slot is simply proportional to the bandwidth realized by the session. The utility is normalized to be at most 1 when the receive bandwidth equals the peak rate. For such linear utility functions the 'marginal utility,' i.e., the derivative of utility with respect to QoS, i.e., bandwidth for the elastic case, is a constant. In practice such tasks would typically have a decreasing marginal utility with additional bandwidth, i.e., they value a reasonable amount of bandwidth, and thus reduced delay, but see diminishing returns as it becomes sufficiently high.

The second simple option for an elastic utility is to use an exponential function

$$u_i(x) = 1 - e^{-\alpha_i x},$$

where $\alpha_i > 0$. As shown in Figure 1 this is a strictly concave increasing function of the bandwidth x saturating at 1. Here, α_i might be selected to be smaller for tasks which respond poorly to low bandwidth or high overall transfer delays.

A typical elastic task would be a file transfer mediated by the TCP protocol[1]. The utility of such a task primarily depends on the delay to transfer the file or equivalently on the average bandwidth prior to completion. An elastic application that must be expedited should either be assigned a higher weight or

[1]Note that TCP is a reactive protocol, it slows down the rate of transmission when there is congestion or delays and speeds up otherwise.

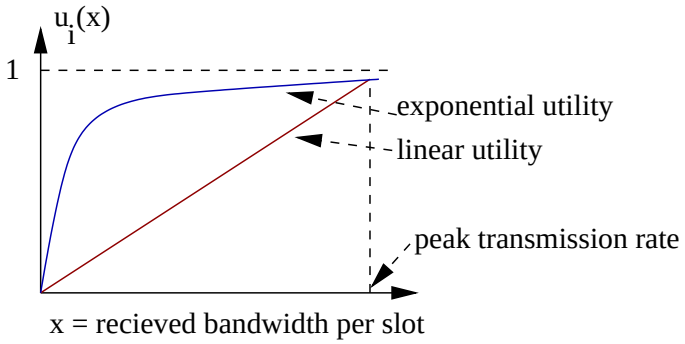

Figure 1: Examples of elastic utility functions.

may be assigned a higher sensitivity to the bandwidth it will see on each slot, e.g., lower value of α_i. The above provides some latitude to an adaptation/reconfiguration mechanism to make such tradeoffs which discriminate among the importance and requirements of various tasks.

A final remark. We have defined the normalized current utility of the system to be 1 if there are *no* active flows in the system. Thus suppose an adaptation/reconfiguration system 'blocks,' i.e., allocates no resources to, an elastic flow that is initiated at time s_i. According to our definition of the system utility, this task would then not complete prior to the end of the mission t^*. 'Soft' blocking of an elastic task would contribute a weighted utility of zero to the normalized current utility, Eq. (2), and do so during the period $[s_i, t^*]$, thus reducing the overall system utility Eq. (3). In other words it is preferable to complete such tasks as soon as possible.[2] Indeed since we have normalized the utility of a task to be at most 1. If a task is blocked the overall system utility will degrade. Thus a utility maximizing policy will be driven to complete elastic tasks as soon as possible.

Application-level utility functions–inelastic case. By contrast to elastic tasks the utility of an inelastic task for a given time slot depends in a more complex measure of the QoS it receives. Recall that the inelastic applications of interest in our context are situational awareness, voice, video and collaborative white boards.

For such applications we shall assume that QoS roughly depends on the fraction of packets that are lost or excessively delayed.[3] For simplicity we shall define the QoS metric for receiver r of task i at time t as

$$q_i(t,r) = \text{fraction of 'useful' packets successfully received by } r \text{ in time slot } t. \tag{5}$$

If no packets are transmitted on the t^{th} slot then we define $q_i(t,r)$ to be 1. The 'usefulness' of a packet may depend on its delay, e.g., if a packet is delayed beyond an application dependent bound d_i^* then it is deemed useless. Also if a packet arrives out of order it may be deemed useless for some applications, e.g., a situational awareness update. Note that we will assume that the time slot window Δ is quite a bit larger than the acceptable end-to-end delays of applications otherwise the above QoS metric would be difficult to assess.

The tolerance to loss and delay of various applications may vary. For example smoothing buffer(s) at a receiver end node can buffer packetized streams allowing for a higher or lower tolerance to delays for audio/video streams. Also for media streams lost packets may be concealed using interpolation without substantially degrading performance. We shall let the utility of an inelastic application be a sigmoidal function of $q_i(t,r)$, e.g.,

$$u_i(x) = \frac{1}{1+e^{-\alpha_i(x-\beta_i)}} \text{ where } \alpha_i, \beta_i > 0. \tag{6}$$

[2] One might specify a maximum transfer delay, say m_i for such sessions beyond which the transfer has no value in which case a soft blocking of such a session would penalize the network only for a period $[s_i, \min[s_i + m_i, t^*]$.

[3] Note that delay jitter also impacts QoS, we will however ignore it for now, assuming that absolute delays are sufficiently constrained and playback buffers can smooth out such variations.

As shown in Figure 2 the value of α_i impacts the task's sensitivity to QoS degradation while the value of β_i captures an acceptable 'region of operation'. Thus if too many packets are lost the QoS is unacceptable but once the the fraction of packets delivered is high enough the utility saturates exponentially to 1 at a speed which depends on α_i. Such saturating functions are typical of both empirical and theoretical rate distortion functions associated with voice/video streams.

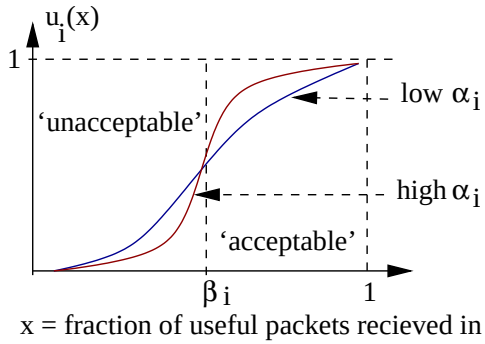

Figure 2: Example of an inelastic (sigmoidal) utility function and it's sensitivity to its parameters α and β.

As with elastic tasks an inelastic task could be assigned insufficient resources leading to 'soft blocking,' i.e., the task remains in the system accruing a zero utility until termination. Thus for example if an inelastic task is given no resources from the start it would contribute a zero term to the weighted normalized current utility, Eq. (2), and to the overall system utility Eq. (3) during the period $[s_i, f_i]$.

2.3.2 Application-level utility functions for multi-point sessions

As mentioned earlier most tasks in a tactical mission will involve sets of receivers. We let R_i denote the set of possible receivers associated with task i. In order to define a utility per slot for such tasks we need to define appropriate compositions that reflect the utility from the point of view of various receivers.

Decomposition into point-to-point tasks. One approach is to simply subdivide tasks that involve multiple receivers into individual point-to-point tasks and treat these separately. In this case each point-to-point task could be assigned a different weight, capturing source-receiver pairs that are of higher importance. However the various point-to-point tasks might still be coupled via a common transport mechanism, e.g., reliable/unreliable multicast. In this case the utility functions defined above can be applied. Unfortunately this would cause an explosion in the number of tasks, for example an n-way push-to-talk voice connection would $n \times n$ different tasks to be defined with their own weights and possibly different utilities. It some cases it makes better sense to make a composition of the performance seen by all receivers, this would reflect the overall utility of a task, for example the utility of broadcast message may be better defined in terms of the fraction of end nodes that received it on time. We discuss such compositions below.

Composite utilities for many-to-many tasks. We propose to define *receiver-oriented* compositions of the utility of a task involving a set R_i of receivers. If task i is elastic we define a receiver oriented quality of service seen by $r \in R_i$ on slot t as

$$q_i(t,r) = \frac{\text{bits successfully received by } r \text{ on time slot } t}{\Delta} \tag{7}$$

and let $q_i(t,r) = 0$ if no new information was received by receiver r on slot t. Analogously if task i is inelastic we define a receiver-oriented quality of service seen by $r \in R_i$ on slot t as

$$q_i(t,r) = \text{fraction of 'useful' packets successfully received by } r \text{ on time slot } t \tag{8}$$

and let $q_i(t,r) = 1$ if no packets were sent to r by any source. Recall that we define the vector $\vec{q}_i(t) = (q_i(t,r)|r \in R_i)$ characterizing the received QoS for each receiver time slot t.

Let $u_{i,r}()$ to be an elastic/inelastic utility function associated with receiver r of task i capturing the utility it derives from its perceived QoS $q_{i,r}(t)$ on a given slot. We define $u_i()$ to be the overall utility for task i as a composition of that seen by the various receivers. Several methods of composition can be considered. The *arithmetic mean*

$$u_i(\vec{q}_i(t)) = \frac{1}{|R_i|} \sum_{r \in R_i} u_{i,r}(q_i(t,r)) \tag{9}$$

gives a sense of the utility seen from a typical receiver's point of view. Assuming a set of receiver weights $w_{i,r}, r \in R_i$ are specified for task i one can define an *weighted arithmetic mean*,

$$u_i(\vec{q}_i(t)) = \frac{\sum_{r \in R_i} w_{i,r} u_{i,r}(q_i(t,r))}{\sum_{r \in R_i} w_{i,r}} \tag{10}$$

which places more emphasis on some receivers than others. One can also compose the receiver's utilities via a *harmonic mean*, i.e.,

$$u_i(\vec{q}_i(t)) = |R_i| \left(\sum_{r \in R_i} \frac{1}{u_{i,r}(q_i(t,r))} \right)^{-1} \tag{11}$$

which avoids excessively biasing the utility towards receivers that see poor performance, e.g., those that are further away from the source. A final possibility is to make the composite utility the *worst case* utility across receivers, i.e.,

$$u_i(\vec{q}_i(t)) = \min_r \{u_{i,r}(q_i(t,r))|r \in R_i\} \tag{12}$$

which would in principle drive a reconfiguration policy to make the QoS of all receivers roughly equal and as high as possible. Note that if task i is elastic, and transported via an acknowledgment based reliable multicast service, receivers may make progress in lock step. In this case the QoS, seen by various receivers would also be approximately the same, and all of the above would be roughly the same. However negative acknowledgment based reliable multicast service [1] or even peer-to-peer file sharing schemes would not have this property.

2.4 Possible Supplementary Operational Constraints

As mentioned earlier we recommend the capability to supplement the overall utility with explicit operational constraints. These are explicit directives the network designer can use to constrain the reconfiguration/adaptation mechanisms. As with the overall utility function, one must be able to assess whether they are met when the constraints are *feasible*, i.e., it is indeed possible to meet them.

Topology and connectivity. When a MANET reconfigures/adapts to optimize an overall utility in support of a given *independent* workload it will naturally attempt to maintain connectivity/reachability information required to support the workload. Since such workloads are independent of the connectivity the network maintains, a connection which can not be established, e.g., because a destination is not reachable, will simply be lost with the associated impact on the overall utility. Thus our overall utility reflects the need to maintain connectivity, at least that which is appropriate for the given workload, since we assume the precise workload is not known a priori this provides some motivation for the MANET to exert significant effort to maintain and acquire reachability information in a timely fashion. However, in practice the application workload is not 'independent' of the connectivity the MANET maintains. A node which is not known or reachable, is a node to which one could not attempt connecting to. Because maintaining connectivity is costly in terms of energy and overhead traffic, and there are aggressive mechanisms that could be used to achieve this goal, it may still make sense to explicitly require the system meet a specific connectivity constraint.

To that end one can define the current normalized utility function subject to system constraints by simply checking that a constraint is satisfied at each time slot. Thus for example let C_t be a connectivity predicate that should be satisfied at slot t, e.g., the network is connected, then we can define the overall system utility subject to such constraints as

$$U^n(W,p) = \frac{1}{t^*}\sum_{t=1}^{t^*} U_t^n(W,p)\mathbf{1}\{C_t\} \tag{13}$$

where $\mathbf{1}\{C_t\}$ is 1 if C_t is true and zero otherwise. Under this overall utility function whenever the system does not satisfy the desired constraints the current normalized utility is set to zero irrespective of the utilities that various tasks may be achieving. This is a hard constraint on operation, but one may define soft constraints whereby partial satisfaction multiplies the current normalized utility by a number between 0 and 1 capturing the degree of violation.

Prioritization and preemption. Given a set of applications sharing network resources a basic objective for the network designer will be to prioritize access in the sense of deciding which applications should be given a larger share of limited resources. With the current utility function the network designer can assign a large weight w_i to a task i he wishes to favor. Unfortunately this is only a relative prioritization whose impact depends on the resources required, current load, types of competing tasks, etc. Thus the precise impact of such weights is hard to evaluate a priori. As such it may be desirable to give the system designer additional flexibility which allows him to specify preemption rules that should be followed. For example, he might wish to indicate that task (or application type) i should strictly preempt task j. An adaptation/reconfiguration policy would then operate subject to strictly satisfying such preemption constraints.

One way to enforce such prioritization rules in the overall system utility function is to define a predicate $P_{t,i}$ or preemption rules that must be satisfied by tasks that are concurrently active. The current normalized utility at time t subject to such constraints would then be given by

$$U_t^n(W,p) = \frac{\sum_{i\in A_t} w_i\, u_i(\vec{q}_i(t))\ \mathbf{1}\{P_{t,i}\}}{\sum_{i\in A_t} w_i} \tag{14}$$

where $\mathbf{1}\{P_{t,i}\}$ is 1 if prioritization/preemption rules for task i are satisfied on slot t and zero otherwise. Thus violating the prioritization predicate in allocating resources would seriously penalize an adaptation and/or reconfiguration mechanism. Such prioritization rules would constrain the tradeoffs the system can make and thus the behavior of reconfiguration and/or adaptation schemes. The overall system utility would be defined as before possibly including the additional connectivity constraints defined above.

3 System Utility as a Test and Evaluation Metric

As mentioned in Section 1 the system utility metric serves to drive adaptation/reconfiguration as well as a measure of how well a particular policy is performing relative to another. In other words it serves as a test and evaluation metric for various solutions to the adaptation/reconfiguration problem relative to the designer's objectives as implicitly captured by the overall system utility function. A test in which a policy achieves a high normalized overall utility close to 1 is one in which the system was able to carry the workload offering a high QoS to all tasks and effectively realizing the tradeoffs associated with the utility function. We note that we have normalized our metric, so one should only compare the overall utility achieved by the system for the *same* workload.

Evaluating the overall system utility requires monitoring the set of active tasks and the QoS realized for the receivers associated with each task. Recall that the system dynamics have been discretized so that the QoS seen by receivers is summarized on a Δ sec interval. We have defined a receiver-oriented overall utility function whence each receiver needs to be able to report its perceived QoS over such intervals, e.g., for elastic applications the receiver would report the bandwidth seen on the slot, while for inelastic applications

a receiver would report the fraction of 'useful' packets it received. Recall that for inelastic traffic usefulness may depend on the application type. In addition per slot utility functions and weights are associated with each task/receiver, which translate the perceived QoS to receiver utility, and which in turn can be composed, to evaluate the overall current normalized system utility Eq. (1) and overall normalized system utility Eq. (3). The adaptation/reconfiguration system may wish to have direct access to the realized QoS vectors, and tasks utility functions, for adaptation. However for test and evaluation purposes the overall normalized system utility function will summarize how well the system is performing.

4 Related Work

There has been a substantial amount of related work that is relevant in considering the development of appropriate user and system utility functions for wired and wireless systems. It would be unreasonable to attempt to provide a complete bibliography of this area. In this section we provide a brief introduction highlighting key papers and research that might be appropriate towards deepening some of the considerations we have addressed in this paper.

The notions of elastic and inelastic application types used in this paper were perhaps first introduced in the networking context by [2]. Experimental support for modeling the user perceived utility for data transfer applications as an elastic utility, i.e., concave function, can be found in [3]. For inelastic, e.g.,, multimedia traffic, experiments [4, 5, 6, 7] have indeed suggested the existence of a threshold beyond which the user does not perceive improvement in QoS.

At the core of the adaptation/reconfiguration problem is the question of how resources should be shared among ongoing sessions. This has been the center of a rich research activity in recent years. The basic paradigm is the definition of a typically *convex* optimization problem associated with a fixed set of tasks which have elastic utility functions in their allocated rates. Much of this work has focused on investigating how to best perform congestion control, or indeed to understand the implicit utility function associated with TCP congestion control. As mentioned earlier there is indeed much work in this area – a representative paper dealing with elastic traffic would be [8], more recently work addressing inelastic tasks would be [9] and the joint congestion and power control in wireless systems [10]. When the network supports inelastic tasks, or both elastic and inelastic tasks, the associated utility maximization problem may no longer be convex, making the optimization task at hand much more complex.

5 Conclusion and Further Research

This paper develops the elements of a possible overall system utility function to drive adaptation/reconfiguration of MANETs. We have kept things simple by distinguishing only among elastic and inelastic application types, but have introduced some complexity to handle multi-point tasks which are likely to be common in tactical missions. Perhaps the most subtle aspect of the proposed framework is the need to specify an overall utility function that appropriately captures the utility of a system subject to a *dynamic* workload, i.e., were tasks come and go. To that end we proposed a normalized overall current system utility which is averaged over time so as to capture the overall normalized system utility. A key aspect of our definition is that the current overall utility of the network is set to 1, i.e., the highest possible, when the system is not currently supporting a task. Thus when tasks are initiated the job of the adaptation/reconfiguration policy is to adjust resources etc, so as to expedite elastic flows, i.e., return to the utility 1 state, and/or give inelastic flows the highest possible utility. Another key aspect is that such tradeoffs should be guided implicitly by the weights and application level utilities in the overall system function.

There are many interesting research and implementation issues associated with the problem we have discussed in this paper. We briefly comment on two of these. First, for simplicity in Section 2.1 we formally

defined the notion of an independent task workload, where the same sequence of tasks is offered to the MANET irrespective of the QoS seen by each task or the network topology. Making this assumption seems reasonable both to understand and develop adaptation/reconfiguration policies as well as from the perspective of testing and evaluating such technology. However in practice there may be dependencies among tasks, e.g., if one fails, or sees poor performance, the subsequent task may not be initiated. Future work might look at explicit usage patterns, i.e., threads of tasks, and define a higher level utility function associated with threads of tasks. The second issue is addressing the challenge of defining the 'right' mission utility function in the first place. In particular while our weighted additive utility function is quite intuitive and would roughly drive the system in the desired direction, in some scenarios it may not be the best. Specifically the precise impact of the relative weighting among tasks is difficult to evaluate a priori – it depends on a variety of scenario dependent characteristics, e.g., available and required resources, load and contending task types etc. As such it would be interesting to close the loop with an automated mission understanding subsystem which based on additional feedback from unsatisfied users adjusts the weights on an overall system utility to better serve the current or subsequent missions. We see this as an additional level of flexibility that one might eventually study once succesfull reconfiguration/adaptation policies for a given overall system utility are devised.

References

[1] A. Adamson, C. Bormann, M. Handley, and J. Macker. Negative-acknowledgmenet (nack)-oriented reliable multicast (norm) protocol. *Networking Working Group RFC 3940*, Nov. 2004. www.ietf.org/rfc/rfc3940.txt.

[2] S. Shenker. Fundamental design issues for the future internet. *IEEE J. Select. Areas Commun.*, 13(7):1176–88, Sept. 1995.

[3] Z. Jiang et. al. A subjective survey of user experience for data application in future cellular wireless networks. *Proc. of SAINT*, 2001.

[4] G. Ghinea and J.P. Thomas. Qos impcat on user perception and undestanding of multimedia video clips. *ACM Multimedia*, pages 49–54, 1998.

[5] J.Lu et. al. Measuring ATM video quality of service using an objective picture quality model. *Proc. of SPIE*, 3845:290–297, Nov. 1999.

[6] I. Dalgic and F.A. Tobagi. Glitches as a measure of video quality degradation caused by packet loss. *Proc. 7th International Conference on Packet Video*, pages 201–206, 1996.

[7] J. Zamora et. al. Objective and subjective quality of service performance of video on demand in ATM-WAN. *Signal Processing - Image Communications*, 14(6-8):636–654, 1999.

[8] F.P. Kelly, A.K. Mauloo, and D.K.H Tan. Rate control in communication networks shadow prices, proportional fairness and stability. *Journal Oper. Res. Soc.*, 15(49):237–55, 1998.

[9] M. Chiang, J.W. Lee, R. Calderbank, D. Palomar, and M. Fazel. Network utility maximization with inelastic, coupled and rate reliability tradeoff utilities. 2004. Preprint.

[10] M. Chiang. Balancing transport and physical layers in wireless multihop networks: jointly optimal congestion control and power control. *IEEE JSAC*, 23(1), Jan. 2005. To appear.